7N-13-TM
91480
P-7

Knowledge-Based Process Control and Diagnostics for Orbital Cryogen Transfer

Eric A. Raymond

Technical Report FIA-89-10-03-01
October 1989

NASA Ames Research Center
Artificial Intelligence Research Branch
Mail Stop 244-17
Moffett Field, CA 94035-1000

KNOWLEDGE-BASED PROCESS CONTROL AND DIAGNOSTICS FOR ORBITAL CRYOGEN TRANSFER

Eric A. Raymond
Sterling Software

NASA Ames Research Center
Moffett Field, CA 94035
raymond@pluto.arc.nasa.gov

Abstract

AFDex (\'af-dek\) is a rule based system designed to provide intelligent process control, diagnosis, and error recovery for a Shuttle based cryogenic experiment, SHOOT (Superfluid Helium On-Orbit Transfer). This paper describes the AFDex system in the context of traditional associative, model-based, and qualitative systems and discusses the implications of this first expert system in space.

Introduction

AFDex (\'af-dek\) is a rule based system designed to provide intelligent process control, diagnosis, and error recovery for a Shuttle payload bay cryogenics experiment, SHOOT (Superfluid Helium On-Orbit Transfer). The mission's goal is to acquire science data on the properties of superfluid helium in micro-gravity and demonstrate critical technology required to service cryogenically cooled satellites in a microgravity environment.

AFDex is one of four programs which will operate on the Shuttle's Aft Flight Deck (AFD) 80386-based GRiD laptop computer (also referred to as the Payload and General Support Computer, PGSC). The other three programs are used in conjunction ground software to provide crew members with real-time, graphical feedback from the SHOOT payload during specific phases of the mission [1]. AFDex is designed to control the payload in the absence of ground support and expert operators.

The experiment consists of a series of transfers between two dewars in the shuttle cargo-bay. Each dewar (Figure 1.) is heavily instrumented with a variety of devices. *Commandable* devices perform specific actions (i.e. open valve, measure resistance) in response to user directives. *Observable* devices provide scientific and engineering data from the payload and its support electronics (i.e. pressure, power load). *Passive* devices (i.e. phase separators) act in response to changes in the dewar environment, but are not commandable nor directly observable.

A hierarchy of controllers provide the interface to the payload. A Command & Data Handling Unit (C&DH) is responsible for routing information from/to the AFD computer to/from the lower level avionics: the Heater, Level, & Valve System (HLVS), the Temperature & Pressure Measurement System (TPMS), and the Power Distribution Unit (PDU). Each of these boxes is responsible for a particular subset of the commandable and observable devices. Redundant units enable a degree of recovery from faults at this level. (Figure 2.)

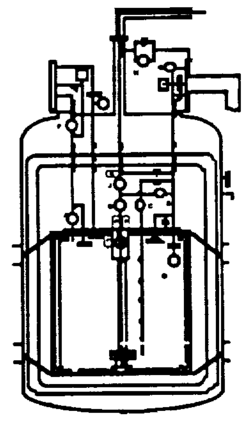
Figure 1. Instrumented Dewar

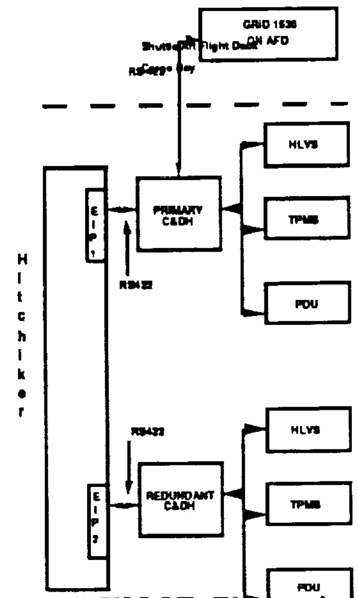

Figure 2. SHOOT Electronics

Description

A SHOOT mission objective is to perform cryogenic transfer of superfluid helium in the absence of ground support. The complexity of the task requires the application of knowledge comprable to that of the experiment's principal investigator and support staff. This knowledge could be provided by a highly trained crew member who would operate the payload much in the same manner as the ground system's operators. Given the complexity of the project, the overhead of crew training, and limited in-flight crew resources, this is not a viable alternative. Furthermore this approach would not scale to provide adequate support for future missions requiring routine, multi-hour transfers. AFDex encapsulates this knowledge in an on-orbit system which eliminates crew interaction with the payload at a low level. Furthermore, AFDex provides a level of functionality beyond that of a nominal transfer; the system is capable of diagnosing faults and recovering.

AFDex *controls* and *monitors* the process of a cryogenic transfer. If an anomalous condition arises, AFDex *hypothesizes* a set of plausible diagnoses which are consistent with its current view of the payload. The system then applies additional knowledge in order to *select* a hypothesis and appropriate *recovery* actions.

Figure 3. Valve Failure and User Consultation

Before AFDex selects a hypothesis, it initiates a consultation with the crew by posting a menu to the AFD display. In addition to generic information which is always displayed, specific information related to the current situation and each hypothesis is included with the menu (Figure 3.). If no response is received from the crew, the system proceeds in the absence of external advice. This user interaction is *asynchronous* and *preemptable*. Asynchrony allows telemetry processing and rule firing to occur during interactions (and thus allow the system to continue reasoning about events). Preemptability allows the system to interrupt an interaction in response to events. This mechanism allows the user to interject knowledge at discrete points during execution and thus effectively guide an otherwise autonomous system.

Implementation

AFDex consists of a mixture of C code and CLIPS rules. CLIPS is a forward chaining, RETE production system [2] shell developed by NASA. Written in C, CLIPS is extremely fast and compact and portable. Flow of control in the rule based system is essentially opportunistic and extremely flexible. This facilitates the integration of process control, diagnosis, error recovery, and user interaction.

Although AFDex is a "rule based" system, a significant portion of the system is procedural C code. In general, this is true for most real world applications; at some level (which may be extremely obfuscated) it becomes practical to use traditional programming paradigms. Typically this occurs when interfacing with the external world. In AFDex, the user interface, graphics, and telemetry processing routines were developed in C. Much of this code was cast as an object system. [3]

For example, all telemetry is treated as objects. AFDex receives information from the payload in the form of telemetry packets. Each packet contains data in a variety of formats each of which requires a different form of processing. Each type of data is a class of telemetry objects. Each instance of each type of data in a packet corresponds to an instance of a telemetry object. An instance contains information unique to itself (i.e. graphic objects which track this telemetry device) and information inherited from its class (i.e. conversion routines, storage requirements). Each telemetry object is responsible for reading its raw data from the telemetry stream, storing the data in its object structure, converting the data, passing the data to any inferior objects, updating graphical displays, and asserting information into the knowledge base.

Process Control and Monitoring

In order to perform a successful transfer, AFDex must generate appropriate commands that modify the state of the payload. Additionally, the system must process feedback from the payload to determine success or failure of its commands.and the general health of the experiment.

Process control in AFDex is conceptually simple. Given knowledge of the current state and of a desired state, *rules of achievement* fire to issue commands that incrementally transform the current state into the desired goal state. Alternately, these rules may post a desired state of their own (a subgoal) which in turn will trigger other rules of achievement. Rules of achievement follow from device models (i.e. how a valve works) and operations models (i.e. how to cool down a dewar). The desired state follows from a prior rule of achievement or an initial desired state (80% helium in port dewar). In this model, commanding results from achieving a desired state.

Monitoring is similar in nature to process control. Like rules of achievement, *monitoring rules* are activated as a direct consequence of a difference between a current and desired state. Their action, however, is to initiate diagnosis. Typically, this desired state reflects the anticipated consequences of a prior action. (In some sense, the distinction between these monitoring rules and the rules of achievement is artificial. For example, it is trivial to cast process control as the consequence of corrective actions in response to a diagnosis initiated by a monitoring rule.)

Diagnosis

Once AFDex has determined that a problem exists, it must determine the cause. Faults range from mundane, transient problems with a device or subsystem (i.e. corrupted telemetry), to catastrophic failures (i.e. Castle's catastrophe, a unique phenomena during which superfluid helium vents through an otherwise impermeable phase separation device). Typically a diagnosis will isolate the faulty behavior of a particular device or operational procedure Contrast this approach with many current "diagnosis" system which simply localize the origin of a fault to a device. This facilitates error recovery and will be discussed in the following section.

Diagnosis rules generate all hypotheses consistent with the current state. *Associative* rules generate hypotheses based upon associations between symptoms and the underlying fault. *Model-based* rules generate hypotheses based upon discrepancies/similarities of states predicted by a model and the current state. AFDex does not utilize quantitative models of superfluid helium physics. Rather, the system incorporates functional, qualitative models of hardware behavior and operations.

Given a set of consistent hypotheses, *selection rules* apply additional knowledge to prioritize the set and select a hypothesis for commitment. This knowledge includes a priori preference criteria, histories of device behavior, prior actions, prior hypotheses, and the results of consultations with crew members. The system does not currently support *active diagnosis* whereby the system probes for information which would further discriminate hypotheses.

Commitment entails an assignment of blame to a single hypothesis in exclusion of all other hypotheses. *Multiple faults* are supported only as a special case of a single hypothesis. The entire diagnosis mechanism prior to a commit is *preemptable*. That is, if the problem is rectified or a problem of greater importance arises, the system may ignore or suspend the current diagnosis.

Error Recovery

Once the system has committed to a hypothesis, it attempts to recover from the diagnosed failure. Due to the specificity of diagnoses generated by the system, there is typically a 1:1 mapping between diagnosis and error recovery scripts. Scripts provide a mechanism for specifying a sequence of actions whose parameters become instantiated based upon context. Scripts include actions such as commanding devices, modifying models, and selecting desired states. Since these actions typically have monotonic side effects, preemption of error recovery is limited to a few highly critical events.

Errors fall into one of four classes. *Spurious* errors are those which rectify themselves without external influences. Recovery typically involves waiting longer or repeating a previous action. *Locally recoverable* errors involve a partial failure of a device whereby an alternate action for that device will resolve the problem. *Globally recoverable* errors require an alternate plan for achieving a desired goal state (i.e. use alternate flow path if a valve is stuck). *Nonrecoverable* errors cannot be resolved by any action. The only alternative is to either ignore the situation or terminate the transfer.

Related Work

Associative Diagnosis generates hypotheses based upon associations between symptoms and the underlying fault. *Model Based Reasoning* (MBR) generates hypotheses based upon discrepancies/similarities of states predicted by a model and the current state. Although a liberal interpretation of MBR requires only the use of a model-based representation, traditionally the term describes systems which reason based upon *device models* and the *structure* by which devices are connected to one another. Typically the structure of the system drives the reasoning [4]. AFDex incorporates both associative diagnosis and reasoning based upon device models. Unlike traditional MBR, AFDex does not rely upon the underlying structure of its domain to guide its reasoning.

Qualitative Physics reasons about the physical world by representing continuous properties by discrete systems of symbols [5]. AFDex reasons at both the qualitative and quantitative levels. In many cases, this reflects inherent qualitative properties of a device (i..e. valve states). Qualitative representation is used when it improves the effectiveness and efficiency of reasoning. Quantitative representations are used when loss of information associated with a qualitative abstraction is not tolerable.

Real Time Issues

AFDex interacts with the SHOOT payload in real time. "Real Time" is defined as an empirical measure of a system's ability to respond to events (i.e. telemetry) within a given time constraint. Although provable guarantees of response time are desirable, most systems opt for empirical, ad hoc demonstrations that the system performs within specifications when responding to a number of test cases. Most systems which provide a formal basis for guaranteed response time do so at the expense of expressiveness and functionality [6].

Laffey, et. al. [7] identify a number characteristic key features of real time expert systems. We discuss each feature in the context of AFDex:

Nonmonotonicity: AFDex reasons with data that changes/disappears with time.

Continuous Operation: Although not strictly continuous (the transfer has a well defined end state), AFDex addresses the key issues: the ability to function in the presence of system failures and efficient, non-obtrusive garbage collection

Asynchronous Events: AFDex is capable of handling unexpected events and prioritizing events of varying importance.

Interface to External Environment: AFDex processes data and issues command via telemetry. Additionally the system supports user intervention through a menu-based interface.

Uncertain or Missing Data: AFDex supports this as a special case of diagnosis and error recovery.

High Performance: AFDex should provide excellent response time to events (on the order of hundreds of milliseconds after arrival of data). This is a soft constraint as there is typically no need to respond faster than on the order of seconds. A number of mechanisms have been developed to improve the throughput of the system.

Temporal Reasoning: AFDex currently deals with time on an alarm basis. That is, events may be triggered based upon a time limit being exceeded. The system also reasons about future (desired events) and maintains histories of past events.

Focus of Attention: AFDex uses a number of methods to focus its attention and thus improve performance. These are discussed below.

Guaranteed Response Times: AFDex responds to events by the time a response is needed.

Integration with Procedural Code: AFDex is written in C and CLIPS. C code is callable from CLIPS rules or executed between rule firings to perform telemetry processing and maintain the user interface.

The RETE pattern/join network in AFDex is driven by facts which are asserted into the knowledge base. This constitutes approximately 90 percent of the time of RETE execution [8]. In order to improve performance, it is necessary to avoid asserting large amounts of irrelevant information, to perform appropriate processing prior to assertion, and to limit the matching of subpatterns within rules which are not applicable.

Filters are used to focus attention onto/away from portions of telemetry. At the lowest level, it is possible to command portions of the avionics as to what data is desired although this method is not generally applicable, is inflexible, and involves non-trivial operation overhead. A more flexible approach is to allow the expert system to selectively control which classes of information are of relevance and should be allowed to assert facts into the knowledge base. AFDex provides a set of CLIPS rule actions that control the assertion of data from telemetry objects into the knowledge base.

Since the pattern matcher only responds to new events yet maintains knowledge of old events, it is not necessary to assert a fact every time a telemetry packet is processed. One method is to assert facts only when a value changes and otherwise assume a constant value. This can easily be extended to incorporate thresholds of change (or other numeric constraints) which must be satisfied before a value is considered to change. A related method is to process the quantitative data and assert a qualitative value.

In order to limit matching of subpatterns within inapplicable rules, a focus of attention mechanism is used within the rule structure. Each rule is prefaced with a pattern describing its applicable context. This context is disjoint from the state information acquired from telemetry. A set of control rules change this context as the system executes. These context patterns effectively guard the remainder of the conjuncts from matching against knowledge in the system unless the context is active.

Another useful mechanism is to change the relative priorities of rule execution and telemetry processing in relation to current load on the system. As the agenda of unexecuted rules grows, the system can increase the rule execution priority by incrementing the ratio of rule firings per telemetry processing. Similarly, as the size of the telemetry buffer grows it is possible to bias priorities in favor of telemetry processing.

Status

The current system is capable of controlling a nominal transfer operation. It has knowledge concerning command generation and receipt, valves, germanium resistance thermometers, and general transfer operations. The system has been tested with simulation at various levels and with the current brassboards of the various avionics subsystems. Unfortunately a more complete system is not available at this time. Tests with an operational flight dewar and support subsystems have been planned.

Conclusion

AFDex is a hybrid system which incorporates concepts from a variety of paradigms as appropriate: production systems, object oriented programming, associative diagnosis, model-based reasoning, and qualitative reasoning. Each paradigm offers a variety of tools. Only those which were appropriate given our real-time constraints were incorporated into the system. We argue that these paradigms are maturing into conventional tools and are being applied to large-scale systems. AFDex demonstrates that rule-based systems have become a mature technology which can be integrated into aerospace operations.

The system described in this paper addresses the requirement of providing "intelligent" control on-orbit for the operations of a specific payload. It also happens to be the first planned use of an expert system in space. This latter concept is largely irrelevant in that our approach would not have differed had the AFD computer been located on Earth The important issue is that new technology is being used in innovative ways to provide functionality that otherwise would have been unachievable.

Acknowledgements

I would like to thank Monte Zweben, David Thompson, and, Tim Castellano for their comments, the entire SHOOT team for their support, and YABIS for providing the motivation to build real systems such as AFDex.

References

[1] Castellano, T.P., Raymond, E.A, Shapiro, J.C., et. al., "Knowledge Based and Interactive Control for the Superfluid Helium On-Orbit Transfer Project", 1989 Goddard Conference on Space Applications for Artificial Intelligence, NASA CP-3033, May 1989.

[2] Forgy, C.L., "RETE: A Fast Algorithm for the Many Pattern/Many Object Pattern Match Problem", Artificial Intelligence 19, 17-37, 1982.

[3] Bobrow, D.G., et. al., "Common Lisp Object System Specification", X3J13 Document 88-002R, 1988.

[4] Davis R., "Diagnostic reasoning based on structure and behavior:, Artificial Intelligence 24, 347-410, 1984.

[5] Forbus, K.D., "Qualitative Physics: Past, Present, and Future", Exploring Artificial Intelligence, Ch 7, 239-296, Morgan Kauffman, San Mateo, CA, 1988.

[6] Haley, P.V., "Real-Time for RETE", Proceedings of ROBEXS '87, Research Triangle Park, NC, Instrument Society of America., 1987.

[7] Laffey, T.J., et. al., "Real-Time Knowledge-Based Systems", AI Magazine, Spring 1988.

[8] Gupta A. and Forgy, C.L., "Measurements on Production Systems", CMU-CS-83-167, Carnegie Mellon University, Pittsburgh, PA, 1983.

~~**RIA-89-05-16-01**~~

~~*Knowledge Based and Interactive Control for the Superfluid Helium On-Orbit Transfer Project*~~
~~TIMOTHY CASTELLANO, ERIC RAYMOND, JEFF SHAPIRO, FRANK ROBINSON, AND DONALD ROSENTHAL~~
~~May 1989~~

~~NASA's Superfluid Helium On-Orbit Transfer (SHOOT) project is a Shuttle-based experiment designed to acquire data on the properties of superfluid helium in micro-gravity. Aft Flight deck Computer Software for the SHOOT experiment is comprised of several monitoring programs which give the astronaut crew visibility into SHOOT systems and a rule based system which will provide process control, diagnosis and error recovery for a helium transfer without ground intervention, Given present Shuttle manifests, this software will become the first expert system to be used in space. The SHOOT Command and Monitoring System (CMS) software will provide a near real time highly interactive interface for the SHOOT principal investigator to control the experiment and to analyze and display its telemetry. The CMS software is targeted for all phases of the SHOOT project, hardware development, pre-flight pad servicing, in-flight operations, and post-flight data analysis.~~

RIA-89-10-03-01

Knowledge-Based Process Control and Diagnostics for Orbital Cryogen Transfer
ERIC RAYMOND
October 1989

AFDeX is a rule based system designed to provide intelligent process control, diagnosis, and error recovery for a shuttle based cryogenic experiment, SHOOT (Superfluid Helium On-Orbit Transfer). This paper describes the AFDeX system in the context of traditional associative, model-based and qualitative systems, characterizes real-time features of the system, and discusses the implications of this first expert system in space.

~~**RIA-89-10-03-02**~~

~~*Interactive Remote Control for an STS-Based Superfluid Helium Transfer Demonstration*~~
~~JEFF SHAPIRO AND FRANK ROBINSON~~
~~October 1989~~

~~The Superfluid Helium On-Orbit Transfer (SHOOT) experiment is a Shuttle-based demonstration of the technology required to service cryogenically cooled satellites in space. The experiment will be controlled continuously during a four to seven day mission by a scientist at a Payload Operations and Control Center (POCC). The SHOOT Command and Monitoring System (CMS) software, developed on a macintosh II, will provide a near real time highly interactive interface for the scientist to control the experiment and to analyze and display its telemetry. The CMS software will communicate via the NASCOM network with the experiment located in the space shuttle's cargo bay. Important features of the CMS include a graphical point-and click interface making the software very easy to use with only a limited knowledge of the experiment. The design of the software was driven by the user interface which was prototyped completely before implementation, and is general enough to be reusable throughout all phases of the experiment. SHOOT is being managed by Goddard Space Flight Center and the software is being developed at Ames Research Center.~~

REPORT DOCUMENTATION PAGE

Form Approved
OMB No. 0704-0188

1. AGENCY USE ONLY *(Leave blank)*

2. REPORT DATE
Dates attached

3. REPORT TYPE AND DATES COVERED

4. TITLE AND SUBTITLE
Titles/Authors - Attached

5. FUNDING NUMBERS

6. AUTHOR(S)

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)
Code FIA - Artificial Intelligence Research Branch
Information Sciences Division

8. PERFORMING ORGANIZATION REPORT NUMBER
Attached

9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)
Nasa/Ames Research Center
Moffett Field, CA. 94035-1000

10. SPONSORING/MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12a. DISTRIBUTION/AVAILABILITY STATEMENT
Available for Public Distribution
Peter Friedland 5/14/92 BRANCH CHIEF

12b. DISTRIBUTION CODE

13. ABSTRACT *(Maximum 200 words)*
Abstracts ATTACHED

14. SUBJECT TERMS

15. NUMBER OF PAGES

16. PRICE CODE

17. SECURITY CLASSIFICATION OF REPORT

18. SECURITY CLASSIFICATION OF THIS PAGE

19. SECURITY CLASSIFICATION OF ABSTRACT

20. LIMITATION OF ABSTRACT